

# SISTEMI IN TEMPO REALE M

## 1 INTRODUZIONE

UN SISTEMA DI ELABORAZIONE OPERA IN TEMPO REALE SOLTANTO SE FORNISCE I RISULTATI ATTESI ENTRO PRESTABILITI LIMITI TEMPORALI DIPENDENTI DAL CONTESTO APPLICATIVO.

UNA VOLTA DEFINITA UNA APPLICAZIONE IN TERMINI DI PROCESSI COOPERANTI CARATTERIZZATI DA PREFISSATE INTERAZIONI, SPECIFICHE TEMPORALI, ARCHITETTURA DEL SISTEMA (MONOMP o MULTIMP), OCCHIO INDIVIDUARE UN OPPORTUNA STRATEGIA DI SCHEDULING.

### TIPOLOGIE DI SCHEDULAZIONE

OFF LINE : INTEGRALMENTE PIANIFICATA A PRIORI

ONLINE : STABILITA A TEMPO DI ESECUZIONE

GUARANTEED : RISPETTA I VINCOLI TEMPORALI DI TUTTI I PROCESSI

BEST EFFORT : OTTIMIZZA LE PRESTAZIONI MEDIE DELL'INSIENE DEI PROCESSI

PREEEMPTIVE : L'ESECUZIONE DI UN PROCESSO PUO' ESSERE SOSPESA

NON PREEEMPTIVE : IN CASO CONTRARIO

### TIPOLOGIE DI PROCESSO

HARD REAL-TIME : I VINCOLI TEMPORALI DEVONO ESSERE SEMPRE RISPETTATI.

I PROCESSI POSSONO ESSERE PERIODICI O SPORADICI.

SOFT REAL-TIME : I VINCOLI TEMPORALI POSSONO ESSERE DISATTESI IN CONDIZIONI DI TEMPORANEO SOVRACCARICO. I PROCESSI POSSONO ESSERE PERIODICI O APERIODICI.

## 2 SCHEDULING PROCESSI HARD REAL TIME

### PROCESSI PERIODICI SENZA VINCOLI DI PRECEDENZA E SENZA RISORSE CONDIVISE

SISTEMA MONOPROCESSORE CON N PROCESSI  $P_j$  ( $T_j$  PERIODO,  $C_j$  TEMPO MASSIMO DI ESECUZIONE,  $D_j$  DEADLINE).

CONDIZIONE NECESSARIA MA NON SUFFICIENTE AFFINCHÉ UN INSIENE DI N PROCESSI SIA SCHEDULABILE È CHE IL FATTORE DI UTILIZZAZIONE U DEL PROCESSORE NON SIA SUPERIORE A 1.

$$U = \sum_j^m U_j = \sum_j^m C_j / T_j \leq 1$$

### SCHEDULING CLOCK DRIVEN (OFFLINE, GUARANTEED, NON PREEEMPTIVE)

SEMPLICE DA REALIZZARE MA NON IDONEA IN CONTESTI CHE IMPLICANO DINAMICITÀ E FLESSIBILITÀ.

TUTTI I VINCOLI TEMPORALI SONO SODDISFATTI A PRIORI (PROBLEMA NP HARD).

LO SCHEDULER PRENDE UNA DECISIONE SU QUALE PROCESSO MANDARE IN ESECUZIONE QUANDO RICEVE UN'INTERRUZIONE DA UN TIMER HARDWARE OPPORTUNAMENTE PROGRAMMATO.

UN TIPICO AMBIENTE DI ESECUZIONE DI PROCESSI REAL TIME È QUELLO DEL PLC STRUTTURO IN NUOLO CICLICO : PRE ELABORAZIONI INTERNE, LETTURA INGRESSI, ESECUZIONE PROGRAMMA, AGGIORNAMENTO USCITE.

### APPROCCIO CYCLIC EXECUTIVE

① DIMENSIONAMENTO DEL CICLO MAGGIORE :  $\text{mcm}(T_1, \dots, T_m) = M$

② DIMENSIONAMENTO DEL CICLO MINORE CHIAMATO m. VINCOLI → SEGUE...

- VINCOLI :
- (1)  $M \bmod m = 0$  PER NUMERO INTERO DI  $m$ , ALL'INTERNO DI  $M$
  - (2)  $m \geq C_i, \forall i$  OGNI JOB PUÒ COMPLETARE E INIZIARE LA SUA ESECUZIONE ALL'INTERNO DI UN CICLO MINORE  $m$ .
  - (3)  $m \leq T_i, \forall i$  GARANTISCE L'ESECUZIONE IN OGNI CICLO MAGGIOR DEL NUMERO DI JOB ASSOCIATI A CLASCIUN PROCESSO:  $M_j = M/T_i$
  - (4)  $2m - \text{med}(m, T_i) \leq T_i \forall i$  TRA RELEASETIME E DEADLINE DI UN QUALUNQUE JOB SIA COMPRESO UN CICLO MINORE COMPLETO NEL QUALE IL JOB POSSA ESSERE INTERAMENTE ESEGUITO

NUMERO CICLI MINORI :  $M/m$

SE NON ESISTE UN FEASIBLE SCHEDULING  $\Rightarrow$  PARTIZIONAMENTO DEL PROCESSO/I CON MAGGIOR TEMPO DI ESECUZIONE IN SOTTO PROCESSI.

### SCHEDULING PRIORITY DRIVEN

AD OGNI PROCESSO È STATICAENTE O DINAMICAMENTE ASSOCIATA UNA PRIORITÀ IN DIPENDENZA DEI CORRISPONDENTI REQUISITI TEMPORALI. AD OGNI PROCESSO È ASSOCIATA DINAMICAMENTE UN'INFORMAZIONE CHE NE IDENTIFICA LO STATO AI FINI DELLA ESECUZIONE: IDLE, READY, RUNNING.  
L'ESECUZIONE DI UN PROCESSO È SOSPESA SE UN ALTRO PROCESSO DI PRIORITÀ SUPERIORE È PRONTO PER L'ESECUZIONE (PREEMPTION).

PRIORITÀ DEL PROCESSO DIRETTAMENTE PROPORZIONALE ALLA CORRISPONDENTE FREQUENZA DI ESECUZIONE  $1/T_i$ .

L'ALGORITMO DI SCHEDULING SI CHIAMA RATE MONOTONIC PRIORITY ORDERING (RMPO).  
SE UN INSIENE DI PROCESSI PERIODICI È SCHEDULABILE CON ALGORITMO DI PRIORITÀ STATICÀ, ALLORA TALE INSIENE È SCHEDULABILE ANCHE CON RMPO.  
SE NON È SCHEDULABILE CON RMPO  $\Rightarrow$  CON NESSUN ALGORITMO CON PRIORITÀ STATICÀ.

LA CONDIZIONE  $U \leq 1$  DIVENTA ANCHE SUFFICIENTE PER LA SCHEDULABILITÀ DI UN INSIENE DI PROCESSI IN RELAZIONE ARMONICA (ES: 25, 50, 100, 200, ...).

**TEST LIU LAYLAND:** CONDIZIONE SUFFICIENTE MA NON NECESSARIA AFFINCHÉ UN INSIENE DI  $N$  PROCESSI SIA SCHEDULABILE CON L'ALGORITMO RMPO:

$$U \leq U_{\text{RMPO}}(N) = N(2^{1/N} - 1)$$

$$\text{PER } N \rightarrow \infty, U_{\text{RMPO}}(N) = \ln 2 = 0,693$$

$$\text{COROLLARIO DI LIU LAYLAND: } U_{\text{RMPO}} = \frac{N}{\prod_i} (1+U_i) \leq 2$$

**TEST DI KVO-NOK:** RIDUZIONE DEL NUMERO DI PROCESSI E APPLICAZIONE DEI CRITERI PRECEDENTI. ESEMPIO:  $P_1: C=5, T=10, C/T=0,5$   
 $P_2: C=5, T=25, C/T=0,2 \quad \left. \begin{array}{l} P_1: C=7,5, T=25 \\ P_3: C=5, T=50, C/T=0,1 \end{array} \right\} C/T=0,3$

**TEST DI BURCHARD:** L'UTILIZZAZIONE SCHEDULABILE DELL'ALGORITMO RMPO È TANTO MAGGIORE QUANTO PIENO I PERIODI DEI PROCESSI SI DISCOSTANO DALLA RELAZIONE ARMONICA. QUESTO VALORE TENDE AL VALORE UNITO 1 NEL CASO DI PROCESSI SENZIPIEMENTE PERIODICI.

$$X_j = \log_2 T_j - L \log_2 T_j \quad \varepsilon = \max X_j - \min X_j$$

$$U_{\text{RMPO}}(N, \varepsilon) = \begin{cases} (N-1)(2^{\frac{\varepsilon}{1-(N-1)}} - 1) + 2^{1-\varepsilon} - 1 & \text{SE } \varepsilon < 1 - 1/N \\ N(2^{1/N} - 1) & \text{SE } \varepsilon \geq 1 - 1/N \quad (\text{LIU LAYLAND}) \end{cases}$$

## TEST DI HAN

UN INSIEME S DI N PROCESSI È SCHEDULABILE CON RMPO SE (CONDIZIONE SOLO SUFFICIENTE) AD ESSO CORRISPONDE UN INSIEME ACCELERATO S' DI N PROCESSI SEMPLICEMENTE PERIODICI E CON FATTORE DI UTILIZZAZIONE  $U \leq 1$ . INSIEME ACCELERATO OTTENUTO RIDUCENDO IL PERIODO  $T_i$  DI UNO o PIÙ PROCESSI. ( $T_1 \circ T_2$  MULTIPLI?).

## ALGORITMO DI AUSLEY NECESSARIA E SUFFICIENTE

LA SCHEDULABILITÀ È GARANTITA SE IL TEMPO DI RISPOSTA DI OGNI PROCESSO NELLE CONDIZIONI PIÙ SFAVOREVOLI NON ECCEDE LA CORRISPONDENTE DEADLINE.

$$R_i^m = C_i + I(R_i^m) \leq T_i \quad i=1\dots N$$

$$I(R_i) = \sum_{\substack{j|P_j > P_i}} \lceil R_i/T_j \rceil \cdot C_j$$

$$P_1 \quad C=5 \quad T=10$$

$$P_2 \quad C=8 \quad T=19$$

$$P_1 = 5 < 10 \quad OK$$

$$P_2 = 8 + \lceil 3/10 \rceil \cdot 5 = 13$$

$$8 + \lceil 3/10 \rceil \cdot 5 = 18$$

$$8 + \lceil 18/10 \rceil \cdot 5 = 18 < 19 \quad OK$$

INTERFERENZA SUL TEMPO DI RISPOSTA  $R_i$  DEL PROCESSO  $P_i$  DERIVANTE DALL'ESECUZIONE DI PROCESSI DI PRIORITÀ SUPERIORE.

ALTRO TEST: CONDIZIONE SOLO SUFFICIENTE  $C_i + \sum \lceil T_i/T_j \rceil \cdot C_j \leq T_i$   
AUSLEY È IL TOP.

## MODELLO PROCESSI SIA PERIODICI CHE SPORADICI

PROCESSI SPORADICI = PROCESSI URGENTI. NUOVO PARAMETRO  $D_j \leq T_j$  CON  $C_j < D_j$ .  
 $D_j = T_j$  PROCESSO PERIODICO.

## ALGORITMO DI SCHEDULING DMPO (DEADLINE NONOTONIC PRIORITY ORDERING)

AD OGNI PROCESSO  $P_j$  È ASSOCIASTA STATIGRANTEMENTE UNA PRIORITÀ INVERSALEMENTE PROPORTZIONALE A  $D_j$ . ALGORITMO OTTIMO SE FALLISCE NESSUN ALTRO ALGORITMO CON PRIORITÀ STATICA DA ESITO POSITIVO.

## ALGORITMO DI AUSLEY RIVISTO: NECESSARIA E SUFFICIENTE

$$R_i^m = C_i + \sum_j \lceil R_i^m / T_j \rceil \cdot C_j \leq D_i$$

$$C_i + \sum_j \lceil D_j / T_j \rceil \cdot C_j \leq D_i \quad (\text{SOLO SUFFICIENTE})$$

$$\sum_j C_j / D_j \leq U_{RMAP} (N) = N (2^{1/N} - 1) \quad (\text{SOLO SUFFICIENTE})$$

## TEST DI LEHOCKY (SOLO SUFFICIENTE)

$$\sum_j C_j / T_j \leq U(N, \alpha) \quad U(N, \alpha) = \begin{cases} N ((2\alpha)^{1/N} - 1) + 1 - \alpha, & 0,5 \leq \alpha \leq 1 \\ \alpha, & 0 \leq \alpha \leq 0,5 \end{cases}$$

$$\alpha = \min \{\alpha_i\} \quad \alpha_i = \frac{D_j}{T_j}$$

## TEST FATTORE UTILIZZAZIONE EFFICACE

✓ PROCESSO IN ORDINE DI PRIORITÀ DECREScente SI CALCOLA:

$$f_j = \left( \sum_{i \in H_m} \frac{c_i}{T_i} \right) + \frac{1}{T_j} \left( C_j + \sum_{k \in H_m} c_k \right)$$

H<sub>m</sub> = PROCESSI PRIORITARI RISPETTO A  
P<sub>j</sub> CON T<sub>i</sub> ≥ D<sub>j</sub>

H<sub>m</sub> = PROCESSI PRIORITARI RISPETTO A  
P<sub>j</sub> CON T<sub>i</sub> < D<sub>j</sub>

E' VERIFICO CHE:

$$f_j \leq U \quad (N = |H_m| + 1, \alpha = \alpha_j) \quad \forall j$$

## ALGORITMO DI SCHEDULING EARLIEST DEADLINE FIRST (EDF)

AD OGNI PROCESSO E' ASSOCIATA DINAMICAMENTE UNA PRIORITÀ TANTO MAGGIORE QUANTO PIÙ INNINENTE E' LA CORRISPONDENTE DEADLINE ASSOLUTA.

CONDIZIONE NECESSARIA E SUFFICIENTE: SOLO PROCESSI PERIODICI

$$\sum_i c_i / T_i \leq U_{EDF} = 1$$

CONDIZIONE SOLO SUFFICIENTE: PROCESSI SPORADICI E PERIODICI

$$\sum_i c_i / D_i \leq 1$$

## TEST DI BARUAH: PROCESSOR DEMAND

N PROCESSI PERIODICI E SPORADICI, U ≤ 1 E SONO ATTIVATI CONTEMPORANEAEMENTE ALL'ISTANTE ZERO. AI FINI DELLA SCHEDULABILITÀ E' SUFFICIENTE VERIFICARE CHE:

$$B_{I_m}^1 \leq 1 \text{ PER PERIODO.}$$

$$B_I^1 = \sum_{i=1}^N \lceil B_{I_{m-1}}^1 / T_i \rceil \cdot c_i \quad B_{I_0} = \sum_{i=1}^N c_i$$

SI FANNO PIÙ ITERAZIONI FINCHE' B<sub>I<sub>m</sub></sub><sup>1</sup> E B<sub>I<sub>m+1</sub></sub><sup>1</sup> SONO UGUALI.

DA B<sub>I<sub>m</sub></sub><sup>1</sup> ≤ 1 PER PERIODO TROVA d = (k-1) T<sub>i</sub> + D<sub>i</sub>; d < B<sub>I</sub>, 1 ≤ i ≤ N, k ≥ 1  
E VERIFICO CHE:

$$C_p(0, t) = \sum_{i=1}^N \left( \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) \cdot c_i \leq t \quad \forall t \in d$$

DEFINITO t\* =  $\sum_{i=1}^N \frac{(1 - D_i / T_i) c_i}{1 - U}$  SE RISULTA t\* < B<sub>I</sub> POSSO CONSIDERARE SOLO

GLI ISTANTI d < t\*. IN GENERALE SI SCEGLIE SEMPRE TRA IL MINIMO DI t\* E B<sub>I</sub>.

### 3 SCHEDULING PROCESSI SOFT REAL TIME

#### SERVIZI IN BACKGROUND

LE RICHIESTE APERIODICHE CARATTERIZZATE DA VINCOLI TEMPORALI DI TIPO SOFT O NON REAL-TIME VENGONO SERVITE SOLO SE NON CI SONO PROCESSI HARD REAL TIME PRONTI PER L'ESECUZIONE. LA STRATEGIA DI SCHEDULAZIONE DEI PROCESSI SOFT REAL TIME E' TIPICAMENTE FCFS (FIRST COME FIRST SERVE) ED INDIPENDENTE DA QUELLA SELEZIONATA PER I PROCESSI HARD REAL TIME.

#### SERVIZIO TRAMITE SERVER A PRIORITÀ STATICÀ

I PROCESSI SOFT REAL TIME VENGONO GESTITI DA UN PROCESSO SERVER HARD REAL TIME.

IL SERVER E' UN PROCESSO PERIODICO DI TEMPO MASSIMO DI ESECUZIONE E PERIODO PREFISSATI. E' SCHEDULATO CON LA STESSA STRATEGIA DEGLI ALTRI PROCESSI A PRIORITÀ Z/TS.

#### POLLING SERVER (PS)

LA CAPACITÀ DEL PROCESSO SERVER PS E' RIPRISTINATA AL VALORE  $C_S$  ALL'INIZIO DI OGNI PERIODO  $T_S$ . E' CONSUMATA DURANTE IL SERVIZIO DI RICHIESTE SOFT REAL TIME. IN ASSENZA DI RICHIESTE LA CAPACITÀ RESTANTE VIENE SCARTATA.

CONDIZIONE SUFFICIENTE:  $U_p + U_s \leq U_{\max}(N+1) = (N+1) \cdot (2^{z/(N+1)} - 1)$

SE IL PROCESSO SERVER HA MASSIMA PRIORITÀ:

$$U_s \leq \frac{2}{(1+U_p/N)^N} - 1$$

#### DEFERRABLE SERVER (DS)

LA CAPACITÀ VIENE RIPRISTINATA AL VALORE  $C_S$  ALL'INIZIO DI OGNI PERIODO  $T_S$ .

LA CAPACITÀ DISPONIBILE VIENE CONSERVATA IN ASSENZA DI RICHIESTE.

LA CAPACITÀ E' PROGRESSIVAMENTE CONSUMATA DURANTE IL SERVIZIO DI RICHIESTE.

CONDIZIONE SUFFICIENTE:  $U_s \leq \frac{2 - (1+U_p/N)^N}{2(1+U_p/N)^N - 1}$

#### PRIORITY EXCHANGE SERVER (PES)

LA CAPACITÀ PUÒ ESSERE ACCUMULATA, OLTRE CHE AL LIVELLO DI PRIORITÀ  $p(P_s)$ , AL LIVELLO DI PRIORITÀ  $p(P_j)$  CHE CONPETE A CIASCUN PROCESSO PERIODICO  $P_j$  CON  $p(P_j) < p(P_s)$ .

LA CAPACITÀ AL LIVELLO  $p(P_s)$  E' RIPRISTINATA AL VALORE  $C_S$  ALL'INIZIO DI OGNI PERIODO  $T_S$ .

LA CAPACITÀ DISPONIBILE AD UN QUALUNQUE LIVELLO DI PRIORITÀ E' CONSERVATA DURANTE L'ESECUZIONE DI UN PROCESSO PERIODICO DI PRIORITÀ NON INFERIORE.

LA CAPACITÀ DISPONIBILE AL MASSIMO LIVELLO DI PRIORITÀ E' CONSUMATA SIA DURANTE IL SERVIZIO DI RICHIESTE SIA IN ASSENZA QUALEORA NON VI SIANO PROCESSI DI PRIORITÀ INFERIORE PRONTI PER L'ESECUZIONE.

LA CAPACITÀ DISPONIBILE AL MASSIMO LIVELLO DI PRIORITÀ, IN ASSENZA DI RICHIESTE MA CON PROCESSI DI PRIORITÀ INFERIORE PRONTI, VIENE TRASFERITA DURANTE L'ESECUZIONE DI CIASCUN PROCESSO AL LIVELLO CORRISPONDENTE DI PRIORITÀ.

IN PRESENZA DI RICHIESTE,  $P_s$ , SE DISPONE DI CAPACITÀ AD UN QUALUNQUE LIVELLO DI PRIORITÀ, HA LA PRECEDENZA RISPETTO AD UN PROCESSO DI PARI PRIORITÀ.

CONDIZIONI SUFFICIENTI:  $U_p + U_s \leq (N+1)(2^{z/(N+1)} - 1)$  - NON A MASSIMA PRIORITÀ

$$U_s \leq \frac{2}{(1+U_p/N)^N} - 1 \quad \text{SERVER A MASSIMA PRIORITÀ}$$

## SPORADIC SERVER (SS)

LA CAPACITÀ DISPONIBILE È CONSERVATA IN ASSENZA DI RICHIESTE.  
È CONSUMATA SOLTANTO DURANTE IL SERVIZIO DI RICHIESTE.

LA CAPACITÀ VIENE REINTEGRATA SOLO DOPO ESSERE STATA CONSUMATA E NELLA MISURA  
IN CUI È STATA EFFETTIVAMENTE UTILIZZATA.

INDICANDO  $t_a$  L'ISTANTE IN CUI SI VERIFICA LA CONDIZIONE  $C_s > 0$  e  $P_s$  ATTIVO (È IN ESECUZIONE)  
UN PROCESSO  $P_s$  CON  $p(P_s) \geq P_s$  ) E CON  $t_d$  L'ISTANTE IN CUI  $C_s = 0$  OVvero  $P_s$  NON ATTIVO,  
L'ENTITÀ RA E RT SONO DEFINITE COME:

$$RA = \text{CAPACITÀ CONSUMATA IN } [t_a, t_d] \quad RT = \max\{t_a + t_s, t_d\}$$

CONDIZIONE SUFFICIENTE :  $U_s \leq (N+1) \left(2^{\frac{1}{1(N+1)}} - 1\right) - U_p$   
 $U_s \leq \frac{2}{(1+U_p/N)^N} - 1 \quad \text{SE } P_s \text{ HA MASSIMA PRIORITÀ}$

## SERVIZIO TRAMITE SERVER A PRIORITÀ DINAMICA

IL SERVER VIENE SCHEDULATO SECONDO LA STESSA STRATEGIA (TIPICAMENTE RQF) UTILIZZATA PER GLI ALTRI  
PROCESSI UN ACCORDO ALLA SUA PRIORITÀ DINAMICA CORRELATA A  $D_s$ .  
 $P_s$  È UN PROCESSO SPORADICO CON FATTORE DI UTILIZZAZIONE  $\leq 1 - U_p$

## CONSTANT UTILIZATION SERVER (CUS)

$P_s$  È PRONTO PER L'ESECUZIONE NON APPENA SI PRESENTA ALL'ISTANTE  $t_a$  UNA RICHIESTA.

INSERISCE LA RICHIESTA IN TESTA AD UNA CODA E DEFINISCE  $C_s = C_{RA}$  e  $D_s = t_{RA} + C_s/U_s$ .  
 $P_s$  VIENE ESEGUITO IN ACCORDO ALLA PRIORITÀ DINAMICA CORRELATA ALLA DEADLINE  $D_s$  SE  $C_s$  VIENE  
CONSUMATA DURANTE IL SERVIZIO.

EVENTUALI RICHIESTE CHE SI PRESENTINO PRIMA DI  $D_s$  VENGONO ACCODATE.

COMPLETATO IL SERVIZIO, LA RICHIESTA VIENE RIROSSA DALLA CODA E  $P_s$  POSTO NELLO STATO IDLE FINO  
ALL'ISTANTE  $D_s$ .

ALL'ISTANTE  $D_s$  SE UNA o + RICHIESTE SONO GIÀ PENDENTI OPPURE ALL'ISTANTE  $t_a$  DI ARRIVO DI UNA

NUOVA RICHIESTA VIENE POSTO:  $C_s = C_{RA}$  e  $D_s = \max(D_s, t_{RA}) + C_s/U_s$ .

SI DICHIARA  $P_s$  PRONTO PER L'ESECUZIONE.

## TOTAL BANDWIDTH SERVER (TBS)

SIMILE A CUS CON UN'UNICA IMPORTANTE DIFFERENZA:  $P_s$  NON VIENE POSTO NELL'STATO DI  
IDLE FINO ALL'ISTANTE  $D_s$ ; UNA VOLTA COMPLETATO IL SERVIZIO, VERIFICA SE CI SONO RICHIESTE  
PENDENTI o NUOVE IMMEDIATAMENTE.

## 4 SCHEDULING DI PROCESSI IN PRESENZA DI RISORSE CONDIVISE

NO AUTOSOSPENSIONE PROCESSO, STRATEGIE PREEMPTIVE, PRIORITY DRIVEN.

### NON-PREEMPTIVE CRITICAL SECTION (PROTOCOLLO NPCS)

NESSUN PROCESSO IN ESECUZIONE ALL'INTERNO DI UNA SEZIONE CRITICA PUÒ SUBIRE PREEMPTION DA PARTE DI ALTRI PROCESSI.

E' SUFFICIENTE ASSOCIARE AD OGNI PROCESSO  $p_i$  UNA PRIORITÀ "CORRENTE"  $p_{i,c}$  (MASSIMA) INDIPENDENTE DALLA SUA PRIORITÀ "NONINALE"  $p_i$ , QUANDO  $p_i$  INIZIA UNA SEZIONE CRITICA. AL TERMINE DELLA SEZIONE SI ASSOCIA DI NUOVO LA SUA PRIORITÀ "NONINALE".

- NON IMPLICA LA CONOSCENZA A PRIORI DELLE RISORSE UTILIZZATE DAI PROCESSI.
- PREVIENE INVERSIONI DI PRIORITÀ INCONTROULATE.
- PREVIENE LA CONCATENAZIONE DI BLOCCHI: UN PROCESSO PUÒ ESSERE BLOCCATO AL PIÙ UNA VOLTA.
- PREVIENE SITUAZIONI DI DEADLOCK.

USATO CON STRATEGIE DI SCHEDULING RRPO o DAPQ, IL MASSIMO TEMPO DI BLOCCO  $B_i$  DI UN PROCESSO  $p_i$  AVENTE PRIORITÀ  $p_i$  È UGUALE AL TEMPO DI ESECUZIONE  $Z_{ik}$  DELLA PIÙ LUNGA SEZIONE CRITICA RELATIVA AD UNA QUALESiasi RISORSA  $R_k$  UTILIZZATA DA UN QUALUNQUE PROCESSO  $p_j$  DI PRIORITÀ  $p_j < p_i$ .

USATO CON STRATEGIE A PRIORITÀ DINAMICA STILE EDF, UN PROCESSO  $p_i$  PUÒ SUBIRE UN BLOCCO DA PARTE DI UN PROCESSO  $p_j$  SE E SOLO SE:

- 1)  $p_j$  È IN ESECUZIONE QUANDO SI ATTIVA  $p_i$ . }  $\Rightarrow D_j \geq D_i \Rightarrow B_i = \max Z_{jk} : D_j > D_i$
- 2) PRIORITÀ NONINALE DI  $p_i$  È MAGGIORE DI QUELLA DI  $p_j$ .

**PRO:** SISTEMA MOLTO SENZUCCHE E EFFICIENTI IN CASO TUTTI I PROCESSI CONPETONO PER TUTTE LE RISORSE CON SEZIONI CRITICHE DI BREVE DURATA.

**CONTRO:** SE UNA RISORSA È OCCUPATA DA UN PROCESSO, TUTTE LE ALTRE RISULTANO INACCESSIBILI E TUTTI I PROCESSI DI PRIORITÀ SUPERIORE BLOCCATI ANCHE IN ASSENZA DI CONDIZIONI CONFLITTUALI.

### PRIORITY INHERITANCE (PROTOCOLLO PI)

LA SCHEDULAZIONE DEI PROCESSI PRONTI È OPERATA IN BASE A PRIORITÀ CORRENTI CHE COINCIDONO CON:

- 1) PRIORITÀ NOMINALE SE  $p_i$  NON DETENGA ALCUNA RISORSA RICHIESTA DA PROCESSI DI PRIORITÀ SUPERIORE.
- 2) LA PIÙ ALTA FRA LE PRIORITÀ CORRENTI DEI PROCESSI DA ESSO BLOCCATI.

AD UN PROCESSO È NEGATO L'ACCESSO AD UNA RISORSA SOLO SE È GIA' OCCUPATA E IN TAL CASO LA PRIORITÀ CORRENTE DEL PROCESSO BLOCCATO VIENE EREDITATA DAL PROCESSO CHE DETIENE LA RISORSA.

- NON IMPLICA LA CONOSCENZA A PRIORI DELLE RISORSE UTILIZZATE DAI PROCESSI.
- PREVIENE INVERSIONI DI PRIORITÀ INCONTROULATE
- NON PREVIENE LA CONCATENAZIONE DEI BLOCCHI: UN PROCESSO PUÒ ESSERE BLOCCATO PIÙ DI UNA VOLTA.
- NON PREVIENE SITUAZIONI DI DEADLOCK.

#### CON SEZIONI CRITICHE NON ANNIDATE:

INDICANDO IL PRIORITY CEILING (TETTO DI PRIORITÀ)  $P_{CK} = \max \{ p_j \mid p_j \text{ USA } R_k \}$ , IL NUMERO MASSIMO DI BLOCCHI (DIRETTI o INDIRETTI) IN CUI PUÒ INCORRERE CIASCUN PROCESSO  $p_i$  DI PRIORITÀ  $p_i$  È DATO DA:

$$m_i = \min (l_i, r_i) \text{ CON}$$

$l_i = \text{Nº DI PROCESSI DI PRIORITÀ} < p_i \text{ CHE CONDIVIDONO RISORSE CON PROCESSI DI PRIORITÀ} \geq p_i$   
 $r_i = \text{Nº DI RISORSE CON } P_{CK} \geq p_i \text{ UTILIZZATE DA PROCESSI CON PRIORITÀ} < p_i$ .

UN UPPER BOUND DEL MASSIMO TEMPO DI BLOCCO  $B_i$  DI CIASCUN PROCESSO  $p_i$  È:

$$B_i = \min \left\{ \sum_{j \in P_j \cap P_i} \max \{ Z_{jk} \mid P_{CK} \geq p_j \}, \sum_{k \in P_{CK} \cap P_i} \max \{ Z_{ik} \mid p_j < p_i \} \right\}$$

## CON SEZIONI CRITICHE ANNIDATE:

LA PRIORITÀ CHE UN PROCESSO ASSUME ALLORCHÉ RIUSCIA UNA RISORSA NON NECESSARIAMENTE COINCIDE CON LA PRIORITÀ CHE IL PROCESSO AVEVA ALL'ATTO DELL'ACQUISIZIONE DELLA RISORSA.

- L'EREDITÀ DELLA PRIORITÀ È TRASITIVA.

UN PROCESSO  $p_i$  DI PRIORITÀ  $p_i$  PUÒ ESSERE BLOCCATO DA UN PROCESSO  $p_j$  CON  $p_j < p_i$  SE  $p_j$  CONDIVIDE UNA RISORSA CON UN PROCESSO CHE HA O PUÒ EREDITARE UNA PRIORITÀ  $\geq p_j$ .

IL MASSIMO TEMPO DI BLOCCO DI OGNI PROCESSO È CALCOLABILE MEDIANTE IL METODO DI RATKUNAR (UTILIZZO DI ALBERI).

## PRIORITY CEILING (PROTOCOLLO PC)

L'ANALOGO AL PROTOCOLLO PI' DA: L'ACCESSO AD UNA RISORSA È NEGATO SE LA PRIORITÀ CORRENTE DEL PROCESSO CHE NE FA RICHIESTA NON È MAGGIORE DEL TETTO DI PRIORITÀ DEL SISTEMA.  $T_{jk} = \max \{ p_{kr} | R_k \text{ IN USO} \}$ .

- IMPLICA LA CONOSCENZA A PRIORI DELLE RISORSE UTILIZZATE DAI PROCESSI (PER POTERNE DETERMINARE IL TETTO DI PRIORITÀ)
- PREVIENE INVERSIONI DI PRIORITÀ INCONTROLLATE
- PREVIENE LA CONCATENAZIONE DEI BLOCCHI.
- PREVIENE SITUAZIONI DI DEADLOCK.

IL MASSIMO TEMPO DI BLOCCO  $B_i$  DI UN PROCESSO  $p_i$  DI PRIORITÀ  $p_i$  È UGUALE AL TEMPO DI ESECUZIONE  $\bar{\tau}_{ik}$  DELLA PIÙ LUNGA SEZIONE CRITICA RELATIVA A QUALUNQUE RISORSA  $R_k$  CONDIVISA DA UN PROCESSO DI PRIORITÀ  $p_j < p_i$  E DA UN PROCESSO DI PRIORITÀ  $\geq p_i$ :

$$B_i = \max \{ \bar{\tau}_{ik} | p_j < p_i, p_k \geq p_i \}$$

## IMMEDIATE PRIORITY CEILING (PROTOCOLLO IPC)

SIMILE AL PROTOCOLLO PC MA QUANDO UN PROCESSO DETIENE UNA RISORSA, LA SUA PRIORITÀ CORRENTE COINCIDE CON IL MASSIMO TETTO DI PRIORITÀ DELLE RISORSE IN SUO POSSESSO.

PROCESSI CON LA STESSA PRIORITÀ CORRENTE SONO SCHEDULATI SECONDO LA POLITICA FIFO.

UNA RISORSA VIENE SEMPRE ALLOCATA AL PROCESSO CHE NE FA RICHIESTA.

- IMPLICA LA CONOSCENZA A PRIORI DELLE RISORSE UTILIZZATE DAI PROCESSI. (COME PC PROTOCOLLO)
- PREVIENE INVERSIONI DI PRIORITÀ INCONTROLLATE
- PREVIENE LA CONCATENAZIONE DEI BLOCCHI
- PREVIENE SITUAZIONI DI DEADLOCK

IL MASSIMO TEMPO DI BLOCCO È UGUALE AL PROTOCOLLO PC.

## IMPLEMENTAZIONE ALTERNATIVA

IL TETTO DI PRIORITÀ DEL SISTEMA È AGGIORNATO OGNI VOLTA VIENE ALLOCATA O RILASCIATA UNA RISORSA. LE PRIORITÀ CORRENTI DEI PROCESSI COINCIDONO CON LE PRIORITÀ NOMINALI. IL PROCESSO PRONTO A MASSIMA PRIORITÀ, SE CONTRADDISTINTO DA UNA PRIORITÀ MAGGIORA DI QUELLA DEL PROCESSO IN ESECUZIONE, PUÒ CAUSARNE LA PREEMPTION SOLO ALLORCHÉ LA SUA PRIORITÀ RISULTI MAGGIORA DEL TETTO DI PRIORITÀ DEL SISTEMA.

## IPC VS PC

**PRO:** SENZ'ESPLICITÀ REALIZZATIVA, RIDUZIONE NUMERO CAMBI DI CONTESTO, ...

**CONTRO:** PROCESSI CHE NON CONDIVIDONO RISORSE (o ACCESSI CONDIZIONATI A RISORSE) POSSONO CONVENUCE ESSERE SOGGETTI A BLOCCHI PREVENTIVI.

## PRE-EMPTION CEILING PROTOCOL

APPLICABILE IN SISTEMI A PRIORITÀ DINAMICA E LIVELLI DI PRE-EMPTION STATICI OVVERO IN SISTEMI IN CUI POTENZIALI CONFLITTI DI ACCESSO ALLE RISORSE CONDIVISE NON CAMBIANO NEL TEMPO.

IL LIVELLO DI PRE-EMPTION  $p_i$  DI UN JOB RIFLETTE LA POSSIBILITÀ CHE ESSO HA DI CAUSARE O SUBIRE PRE-EMPTION DA ALTRI JOBS. SE  $p_j < p_i$ ,  $p_j$  PUÒ SUBIRE PRE-EMPTION DA  $p_i$ .

E' FUNZIONE DELL'ISTANTE DI ATTIVAZIONE DI  $p_i$  E DELLA SUA PRIORITÀ.

ASSEGNAMENTO  $p_i \Rightarrow (p_i \text{ ATTIVATO DOPO } p_j) \text{ AND } (p_i > p_j) \Rightarrow p_i > p_j$

VALE IN QUEI SISTEMI CHE UTILIZZANO STRATEGIA EDF: SE  $D_j > D_i \Rightarrow p_j < p_i$

LA CONDIVISIONE DI RISORSE TRA DUE PROCESSI  $p_i, p_j$  AVANTI DEADLINE  $D_i, D_j$  NON IMPLICA, SE SCHEDULATI CON STRATEGIA EDF, CHE CIASCUNO DI ESSI INCORRA, ALLORCHE IN ESECUZIONE CON MASSIMA PRIORITÀ, IN UNA SITUAZIONE DI BLOCCO IMPUTABILE ALL'ALTRO.

FUNZIONAMENTO SIMILE AL PROTOCOLLO PRIORITY CEILING CON UNA VARIANTE:

L'ALLOCAZIONE DI UNA RISORSA È OPERATA IN BASE AL CONFRONTO DEL LIVELLO DI PRE-EMPTION DEL PROCESSO CON IL "TETTO DI PRE-EMPTION DEL SISTEMA" DEFINITO COSÌ:

$$\pi_s = \max \{ p_k \mid R_k \text{ è in uso} \} \quad \text{con } p_k \text{ È IL MASSIMO LIVELLO DI PRE-EMPTION DEI PROCESSI CHE POSSONO ACCEDERE ALLA RISORSA } R_k.$$

HA LE STESSE PROPRIETÀ DEL PROTOCOLLO PC IN SISTEMI A PRIORITÀ STATICHE.

IL MASSIMO TEMPO DI BLOCCO DI UN PROCESSO  $p_i$  CON  $p_i$  È UGUALE AL TEMPO DI ESECUZIONE DELLA PIÙ LUNGA SEZIONE CRITICA RELATIVA ALL'ACCESSO DA PARTE DI UN QUALUNQUE PROCESSO  $p_j$  CON LIVELLO DI PRE-EMPTION  $p_j < p_i$  AD UNA QUALUNQUE RISORSA  $R_k$  CON TETTO DI PRE-EMPTION  $p_k \geq p_i$ :

$$B_i = \max \{ \tau_{jk} \mid p_j < p_i, p_k > p_j \}$$

## STACK RESOURCE POLICY (PROTOCOLLO SRP)

REGOLE ANALOGHE AL PROTOCOLLO IPC CON UNA UNICA VARIANTE:

L'ESECUZIONE DI UN PROCESSO HA LUOGO SOLO SE:

- HA LA PRIORITÀ MASSIMA FRA I PROCESSI PRONTI E PRIORITÀ MAGGIORE DEL PROCESSO IN ESECUZIONE
- IL SUO LIVELLO DI PRE-EMPTION È STRETTAMENTE MAGGIORE DEL TETTO DI PRE-EMPTION DEL SISTEMA.

STESSE PROPRIETÀ DEL PROTOCOLLO IPC IN SISTEMI A PRIORITÀ STATICHE.

TEMPI DI BLOCCO MASSIMO = STESSO METODO DEL PRE-EMPTION CEILING.

+ È FACILMENTE ESTENDIBILE NEI CASI IN CUI I PROCESSI CONDIVIDANO RISORSE CON PIÙ UNITÀ!!

INDICANDO CON:  $u_k \geq 1 = \text{n}^{\circ}$  UNITÀ DISPONIBILI PER OGNI RISORSA  $R_k$ .

$m_{jk} = \text{n}^{\circ}$  MASSIMO DI UNITÀ  $R_k$  RICHIESTE DA CIASCUNO PROCESSO  $p_j$ .

$n_k(t) = \text{n}^{\circ}$  DI UNITÀ DI  $R_k$  LIBERE NELL'GENERICO ISTANTE  $t$ .

IL TETTO DI PRE-EMPTION È AGGIORNATO A:  $\max \{ \pi_{ik}(n_k(t)) \}$  con

$$\pi_{ik}(n_k(t)) = \max \{ p_j \mid m_{jk} > n_k(t) \}$$

IL MASSIMO TEMPO DI BLOCCO  $B_i$  DI UN PROCESSO  $p_i$  È UGUALE AL TEMPO DI ESECUZIONE DELLA PIÙ LUNGA SEZIONE CRITICA RELATIVA ALL'ACCESSO DA PARTE DI UNA QUALUNQUE DELLA PIÙ LUNGA SEZIONE CRITICA RELATIVA ALL'ACCESSO DA PARTE DI UNA QUALUNQUE PROCESSO  $p_j$  CON LIVELLO DI PREEMPTION  $p_j < p_i$  AD UNA QUALUNQUE RISORSA  $R_k$  CON TETTO DI PREEMPTION MASSIMO  $\pi_{R_k}(0) > p_j$ .

$$B_i = \max \{ z_k \mid p_j < p_i, \pi_{R_k}(0) > p_j \}$$

## ANALISI SCHEDULABILITÀ PER SISTEMI A RISORSE CONDIVISE

### 1) SISTEMI A PRIORITÀ STATICHE

A) CONDIZIONE SUFFICIENTE. INSIEME DI  $N$  PROCESSI PERIODICI ORDINATI PER PERIODO CRESCENTE.  
SCHEDULAZIONE CON RMPO:

$$\forall i \in \{1, \dots, N\}, \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq U_{RMPO}(i) = \begin{cases} 1 & \text{SE PERIODI IN RELAZIONE ARMONICA} \\ i(2^{2i}-1) & \text{ALTRIMENTI} \end{cases}$$

B) CONDIZIONE NECESSARIA E SUFFICIENTE.  $N$  PROCESSI PERIODICI E/O SPORADICI ORDINATI PER DEADLINE RELATIVA CRESCENTE.  
SCHEDULAZIONE DMPO: (USANDO AURSLEY)

$$R_i = C_i + B_i + I(R_i) \leq D_i \quad \forall i$$

$$R_i^m = C_i + B_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{m-1}}{T_k} \right\rceil \cdot C_k \quad i \neq 1, m = 1, 2, \dots$$

### 2) SISTEMI A PRIORITÀ DINAMICA

A) CONDIZIONE SUFFICIENTE.  $N$  PROCESSI PERIODICI ORDINATI PER PERIODO CRESCENTE.  
SCHEDULAZIONE EDF:

$$\forall i, 1 \dots N : \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq U_{EDF} = 1$$

B) CONDIZIONE SUFFICIENTE.  $N$  PROCESSI PERIODICI E/O SPORADICI ORDINATI PER DEADLINE RELATIVA CRESCENTE. SCHEDULAZIONE EDF:

$$\forall i, 1 \dots N : \sum_{k=1}^i \frac{C_k}{D_k} + \frac{B_i}{D_i} \leq 1$$

C) CONDIZIONE NECESSARIA E SUFFICIENTE.  $N$  PROCESSI PERIODICI E/O SPORADICI ORDINATI PER DEADLINE RELATIVA CRESCENTE. SCHEDULAZIONE EDF:

$$\forall i, 1 \dots N : \sum_{k=1}^i \left( \left\lfloor \frac{t - D_k}{T_k} \right\rfloor + 1 \right) C_k + \left( \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) B_i \leq 1$$

PER IL CALCOLO DI  $t$  VEDI "APPROCCIO PROCESSOR DEMAND"!